DOMAIN ADAPTATION VIA DATA AUGMENTATION

A Thesis
Presented to
The Academic Faculty

By

Eric Gastineau

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science, School of Computer Science

Georgia Institute of Technology

May 2020

DOMAIN ADAPTATION VIA DATA AUGMENTATION

Approved by:

Dr. Joy Arulraj, Advisor
College of Computing
Georgia Institute of Technology

Dr. Zsolt Kira
College of Computing
Georgia Institute of Technology

Dr. Calton Pu
College of Computing
Georgia Institute of Technology

Date approved: April 23, 2020

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Joy Arulraj who supervised my thesis and gave me useful advices.

I also wish to express my deepest gratitude to Abhijit Suprem who gave me guidelines and helpful information for the thesis.

This work would not have been possible without all the staff of Georgia Tech who provided me all the equipment I needed for the experiments. I wish to thank all the people whose assistance was a milestone in the completion of this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Deep learning (DL) models require large labeled datasets for training. Practitioners often need to adapt an existing DL model to a different domain. For instance, a practitioner in a company developing autonomous vehicles may need to adapt an object detection model trained over images collected during the day to handle those obtained during the night. Given the challenges associated with curating a labeled dataset for this domain adaptation task, we seek to automatically transform an existing dataset from one domain to another.

This thesis investigates how data augmentation might help cope with the domain adaptation problem. Inspired by other works on domain adaptation [13, 24, 25], we investigate AMOEBA, a domain adaptation system that leverages a generative adversarial network [35] to convert images from one domain to another. By automating this task, it allows a practitioner to effectively train DL models specialized for their target domain. We propose two techniques for improving the efficacy of AMOEBA: (1) data engineering, and (2) postprocessing. Our evaluation of AMOEBA on two domain adaptation problems shows that it is effective in practice. We conclude with a discussion on the limitations of AMOEBA and potential ways to further improve its efficacy.

Chapter 1

INTRODUCTION

There have been many recent breakthroughs in Deep Learning (DL), particularly in the area of object detection (*e.g.* Fast-RCNN [8], Faster-RCNN [22], YOLO [21]). These models require large labeled datasets for training. Manual labeling of such datasets is a time-consuming and expensive task, particularly for object detection algorithms where the human annotator also needs to label the position and size of each object in the image.

Practitioners often need to adapt an existing DL model to different domains. For instance, a practitioner in a company developing autonomous vehicles may need to adapt an object detection model trained over images collected during the day to handle those obtained during the night. We need a dataset that covers diverse environmental conditions (*e.g.* images collected on foggy days, snowy days, rainy days). It is challenging to collect and label such a dataset. A naïve solution would be to train the model on the domain with maximal data (*e.g.* clear days) and then apply this model on all domains. However, this technique does not work since the model is not effective on images that it has not observed during training. This problem is referred to as *domain adaptation* (DA) [30].

Many different techniques have been developed to tackle domain adaptation. There are 3 approaches of DA : Discrepancy-based, Adversarial-based and Reconstruction-based [30]. Some of these methods are supervised [29, 19, 18] and others are unsupervised (and then don't require labeled data) [31, 5, 6]. Among the unsupervised methods, some are based on generative adversarial networks. They use the source domain to generate synthetic data in the

target domain while preserving the labels [13, 16, 25]. These converted samples are then used to train deep learning models.

In this thesis, we present AMOEBA, a system for tackling DA inspired by these generative adversarial-based domain adaptation methods [13, 16, 25]. AMOEBA relies on data augmentation using a generative adversarial network (GAN). In particular, it augments the training data automatically transforms by transforming an existing labeled dataset in a given domain (*e.g.* clear days) to another domain (*e.g.* foggy days). In this manner, AMOEBA constructs a dataset that covers diverse environmental conditions without requiring manual collection and labeling.

AMOEBA leverages CYCLEGAN [35] for converting images across domains. The main advantage of using CYCLEGAN is that it does not require labeled data for training. We evaluate AMOEBA by using the *fake* dataset it constructs to train an object detection model and then test this model on *real* images from this domain. In particular, we examine the efficacy of AMOEBA on two different domain adaptation problems in the Berkeley DeepDrive (BDD) dataset [33]: (1) day to night, and (2) day to fog.

Thesis statement: Data augmentation is a viable technique for tackling the problem of adapting a deep learning model from one domain to another.

The rest of the thesis is organized as follows. §2 illustrates the DA problem. §3 presents an overview of the DL models that we study in AMOEBA. §4 describes the architecture of AMOEBA. §5 and §6 present techniques for improving the efficacy of AMOEBA: (1) data engineering, and (2) postprocessing. §7 and §8 detail the implementation details and experimental results. §9 lists the limitations of AMOEBA. §10 details potential ways to further improve AMOEBA's efficacy.

Chapter 2

MOTIVATION

Manual labeling of datasets is a time-consuming and expensive task. However, we require datasets that cover diverse environmental conditions to train robust DL models. We seek to address this challenge by automating the construction of datasets, thereby eliminating the need for manual collection and labeling of images. In this chapter, we first illustrate the need for DA. We then explain how CycleGAN is a promising technique to automatically transform images from one domain to another.

2.1 Need for Domain Adaptation

A DL model trained on a given domain typically does not work well when used on another domain. This problem arises even when the domains are similar to each other. Consider an object detection model trained on day images from the BDD dataset [33] (*i.e.* Real-Day images). We use a YOLO model in this experiment. We test it on night images from the same dataset (*i.e.* Real-Night images). We compare the model's accuracy against that of a model trained on another set of night images from the same dataset. Table 2.1 shows the AP (Average Precision) score associated with the nine classes in this dataset.

The results are shown in Table 2.1. We observe that the first model exhibits a drop in accuracy across all classes compared to the second model. This highlights the problem of DA. Indeed, even if these 2 domains (day and night) look similar, we can see differences regarding

Table 2.1 Domain Adaptation from Day to Night. The first model is trained on REAL-DAY images from the BDD dataset [33]. The second model is trained on REAL-NIGHT images from the same dataset. We test both models on on another subset of REAL-NIGHT images.

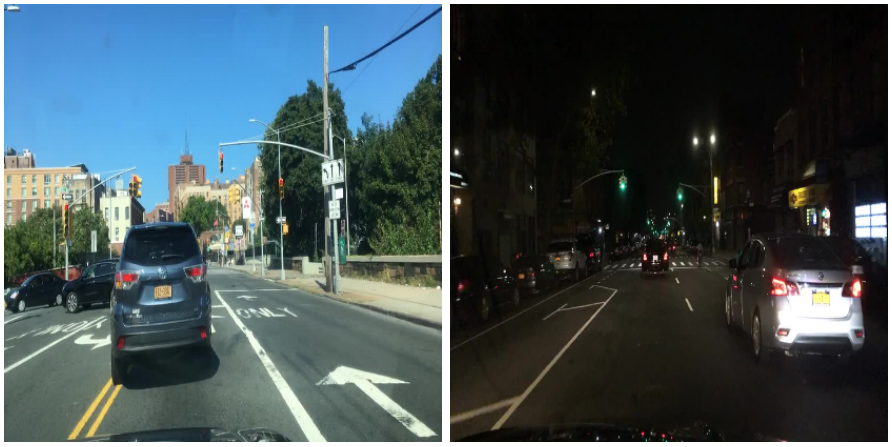
Class	REAL-DAY	REAL-NIGHT	Difference
Bus	0.123	0.199	-0.076
Traffic light	0.273	0.373	-0.1
Traffic sign	0.412	0.427	-0.015
Person	0.278	0.324	-0.046
Bike	0.163	0.234	-0.071
Truck	0.225	0.303	-0.078
Motor	0.055	0.112	-0.057
Car	0.443	0.547	-0.104
Rider	0.091	0.134	-0.043
Overall	0.40	0.484	-0.084

the color distribution (the colors in the night domain are darker) and regarding the number of details (we can hardly recognize trees). We can see these differences in the figure Figure 2.1. We seek to tackle this problem without requiring access to REAL-NIGHT labels.

2.2 Solution: Generative Adversarial Networks

We observe that the information contained in REAL-NIGHT images is lower than that in REAL-DAY images (*e.g.* trees, buildings, vehicles). More generally, images in two different domains need not contain the same level of detail. We leverage this insight in AMOEBA by converting images from a domain wherein images contain more detail to another one with less detail. The details do not change during the night – they are merely less visible. In this thesis, we investigate how to leverage GANs, a new class of DL models, to accomplish this conversion. This solution is inspired by other previous works [13, 16, 25].

We conduct an experiment to examine the efficacy of a particular GAN, CYCLEGAN [35],



(a) Day domain **(b)** Night domain

Figure 2.1 Example of real images in the day domain and in the night domain – Even if these 2 domains (day and night) are similar, there are differences when it comes to the color distribution and the level of details in the pictures.

in transforming images. Figure 2.2a presents an image from [35] that shows the results of a CycleGAN trained to convert photos into Cezanne-styled paintings. This stunning transformation shows the strength of the CycleGAN. Figures 2.2b and 2.2c depict images constructed by AMOEBA using day images. These images look realistic from a human perspective.

(a) Photo to Cezanne-styled Painting

(b) Day to Night

(c) Day to Fog

Figure 2.2 Motivating Examples using CycleGAN– (a) Cezanne-styled painting constructed from a photo. (b) Night image constructed from a day image. (c) Fog image constructed from a day image.

Chapter 3

PRELIMINARIES

In this chapter, we first present an overview of the state-of-the-art DL models for tackling the DA problem. We then describe two DL models that we leverage in AMOEBA: (1) CYCLEGAN, and (2) YOLO.

3.1 Domain Adaptation Techniques

To tackle the paucity of labeled datasets, researchers have designed several techniques for adapting an existing dataset from one domain to another. When the domains are closely related to each other, this problem is referred to as *one-step domain adaptation* [30]. We classify prior efforts for tackling the DA problem into three categories:

- **Discrepancy DA:** This type of DA was the earliest category that was studied. The key idea is to first train a base model on the *source domain*. Then we try to adapt this model with images from the *target domain* in a bid to reduce the distributional discrepancy between the source and target features. There exists a large set of supervised [29, 19, 18] and unsupervised [31, 28, 17] techniques that support this type of DA.
- **Adversarial DA:** The key idea is to train an adversarial network for tackling the DA problem. This category consists of two sub-categories: (1) generative, and (2) non-generative models. With generative models, we convert labeled images from one domain to another and then use these synthetic images to train new models [13, 16, 25, 3]. With non-generative

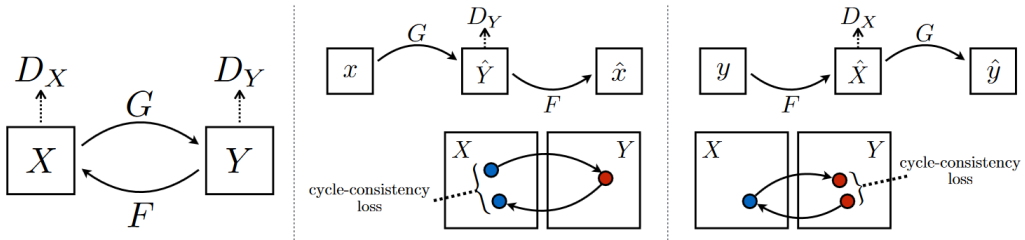


Figure 3.1 Schema of CycleGAN from [35] – CycleGAN converts images from one domain to another.

models, the model is trained using a *domain-confusion* loss aimed to learn a domain-invariant feature space [5, 27, 26]. All of these techniques are unsupervised.

- **Reconstructive DA:** The key idea is to learn a domain-invariant representation that subsumes both domains. We can learn this shared-domain representation either using: (1) encoder-decoder reconstruction [6, 7], or (2) adversarial reconstruction [14, 32].

In this thesis, we focus on a generative adversarial solution which leverages CycleGAN (like [24]) and we try to improve the efficiency of this technique through data engineering and post-processing.

3.2 Cycle GAN

CycleGAN is a recently proposed technique for transforming images from one domain to another [35]. Figure 3.1 illustrates the schema of CycleGAN. Given source and destination domains $\mathcal{X}$ and $\mathcal{Y}$, we seek to find two transformations: (1) $\mathcal{F} : \mathcal{Y} \rightarrow \mathcal{X}$, and (2) $\mathcal{G} : \mathcal{X} \rightarrow \mathcal{Y}$. Besides these two mapping functions, a CycleGAN has two adversarial discriminators D_X and D_Y aimed to distinguish between real images and fake (*i.e.* converted) images.

We train the network by reducing these loss functions:

- **Adversarial Loss Functions:** log D_Y(y), log 1-D_Y(G(x)), D_X(x) and log 1-D_X(F(y)).
- **Cycle-Consistency Loss Functions:** $\|F(G(x)) - x\|$ and $\|(G(F(y)) - y\|$.

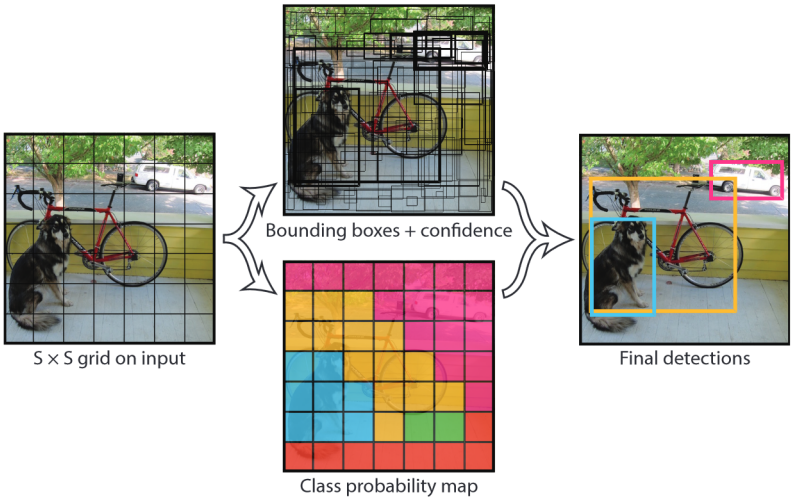


Figure 3.2 Overview of YOLO– The YOLO model detects objects in a given image using a single pass with a convolutional neural network.

We can also use an optional type of loss called identity loss ($\|F(x) - x\|$ and $\|G(y) - y\|$) to regularize the generator. The training is a min-max problem similar to other GANs [11].

3.3 YOLO

With the advent of DL, researchers have presented several algorithms for detecting objects in a given image: R-CNN [9], Fast-RCNN [8], and Faster-RCNN [22]. These algorithms tend to use complex architectures that make them time consuming. YOLO is a recently proposed technique for detecting objects using a simple single convolutional neural network [21]. Figure 3.2 presents an overview of YOLO. It divides the image into a grid and predicts $\mathcal{B}$ bounding boxes and $\mathcal{C}$ class probabilities for each element of the grid. It then reduces the number of selected boxes via non-maximal suppression. It is orders of magnitude faster than earlier object detection algorithms and is still effective.

Chapter 4

SYSTEM OVERVIEW

AMOEBA supports domain adaptation via data augmentation. It transforms labeled images from a domain containing more information to one that contains less information. For instance, in the BDD dataset, it converts day images to night images. AMOEBA leverages CYCLEGAN [35], a cycle-consistent adversarial network, for this conversion.

Figure 4.1 presents the architecture of AMOEBA. It takes two datasets as inputs to train the CYCLEGAN. These datasets need to be representative of their domains to ensure that CYCLEGAN generates realistic images. We first automatically curate a dataset for training CYCLEGAN. In the rest of the thesis, when we train a model on dataset $\mathcal{A}$ and test it dataset $\mathcal{B}$, we refer to this DA task by $\mathcal{A} \rightarrow \mathcal{B}$. Our *baseline* consists of training a model on the source domain $\mathcal{S}$ and testing it on the target domain $\mathcal{T}$ (*i.e.* $\mathcal{S} \rightarrow \mathcal{T}$). We examine the utility of AMOEBA by comparing it against the baseline DA setting (*e.g.*, FAKE-NIGHT $\rightarrow$ REAL-NIGHT as opposed to REAL-DAY $\rightarrow$ REAL-NIGHT). Here, AMOEBA generates the FAKE-NIGHT dataset from the given REAL-DAY dataset. AMOEBA contains an optional *post-processing* component after DA for improving accuracy.

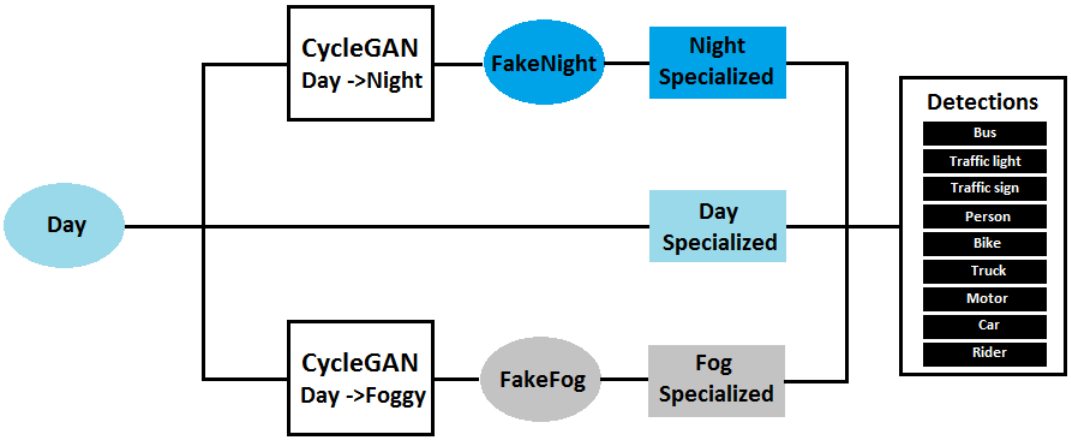


Figure 4.1 Architecture of AMOEBA– CYCLEGAN converts images from the source domain to the target domain (*e.g.* FAKE-NIGHT dataset from the given REAL-DAY dataset). These generated images are then used to train specialized models.

Chapter 5

COPING WITH VISIBLE ARTIFACTS

It is critical to curate datasets that are representative of the source and target domains to train the CycleGAN. We present a technique for manually removing outliers from the dataset in §5.1. We then discuss how to automate this task in §5.2.

5.1 Data Engineering for GANs

After training a first CycleGAN, we found that some images can disrupt the training of the CycleGAN. The goal is to find these "outliers" to suppress them. We manually detect such outliers in the dataset thus:

- We first train the CycleGAN with the given dataset.
- We then look at the resulting converted images and mark the images that are *not realistic* from a human perspective.
- We then remove the images that we deem responsible for these failures from the training dataset and go back to the first step.

Algorithm 1: Data Engineering – Iterative algorithm for removing outliers

Input : CycleGAN Training Dataset $\mathcal{D}$ (*e.g.* REAL-DAY and REAL-NIGHT images),
Test Dataset $\mathcal{I}$ (*e.g.* another split of REAL-DAY images)

Output : Training Dataset $\mathcal{D}'$ (subset of $\mathcal{D}$)

$\mathcal{D}' = \mathcal{D}$;
while *True* **do**
  CYCLEGAN $\mathcal{M}$ = TrainCYCLEGAN ($\mathcal{D}'$);
  Converted Images $\mathcal{I}'$ = $\mathcal{M}$.Convert($\mathcal{I}$);
  if $\mathcal{I}'$ *is realistic* **then**
    exit loop
  end
  Outliers $\mathcal{O}$ = MarkOutliers($\mathcal{D}'$,$\mathcal{I}'$);
  $\mathcal{D}' = \mathcal{D}' - \mathcal{O}$;
end
return $\mathcal{D}'$

Algorithm 1 presents this iterative data engineering technique. `MarkOutliers` is an important function in this algorithm. However, it is an inherently subjective operation based on our perspective.

To illustrate the method, let us consider the converted images illustrated in Figure 5.1. After training the CYCLEGAN with the first BDD dataset (1750 day and night images, respectively), we found that certain converted images suffered from *visible artifacts* (*e.g.* a purple sky with dis-colorations in Figure 5.1a). While going over the training dataset $\mathcal{D}$, we found that many images used for training the CYCLEGAN are taken during dusk, as shown in Figure 5.2. Since there is no strict discontinuity between day and night, these images are tagged as "night" images in the BDD dataset.

We next removed all of these outliers and created a second dataset (1350 day and night

(a) First CycleGAN **(b)** Second CycleGAN

Figure 5.1 Impact of Data Engineering – Images converted using CycleGANs trained on the first and second training datasets, respectively. The second CycleGAN removes the visible artifacts (*e.g.* discoloration in the sky) that are present in the images generated by the first CycleGAN.

images, respectively). We trained the second CycleGAN on this dataset. As shown in Figure 5.1b, the second CycleGAN removes the visible artifacts present in Figure 5.1a. This method of data engineering is not perfect and we still had failures. The problem is that we choose the images to discard only through subjective observations. We use a human eye to "detect" the outliers that may have created these failures. The process can be clumsy. Even if some outliers are obvious, the majority of images can be seen as "normal images" representative of their domain. It is hard to determine a relation between a particular image and a specific failure in the converted dataset. This is why we tried another method which detect the outliers automatically with objective considerations.

5.2 Automation of Data Engineering

We raised the problem of data engineering for the training of the CycleGAN. We would like to have a better method to find outliers that would be automated and that would not be based on

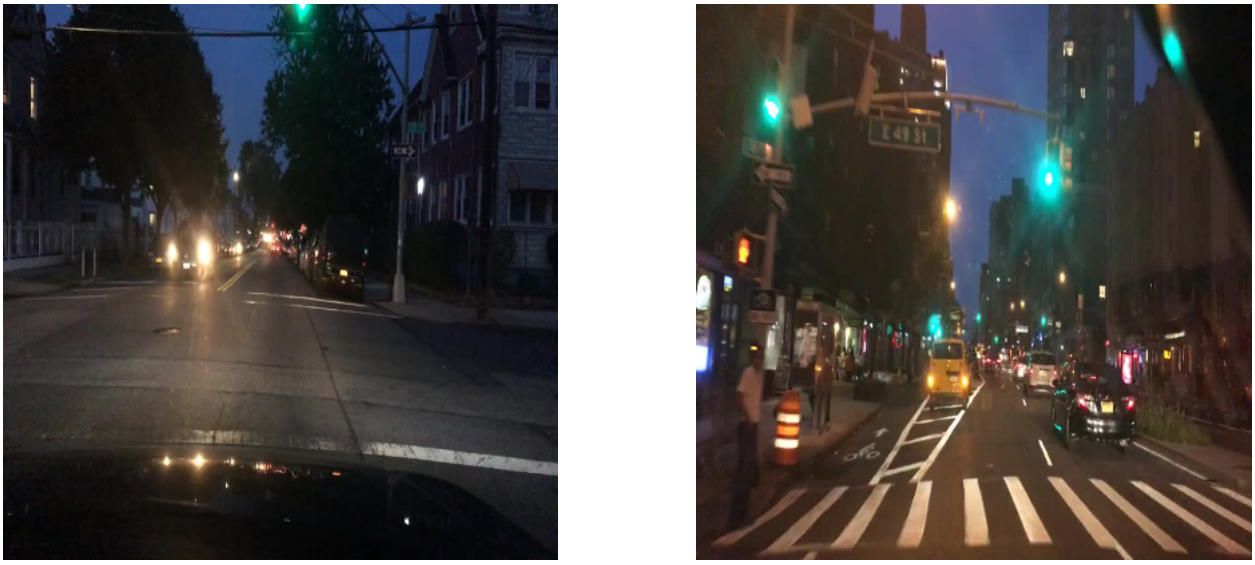


Figure 5.2 Dusk Images – Images in the first training dataset that were not taken during the night, thereby lowering the efficacy of the first CycleGAN.

subjective human observations.

One idea could be to look at the loss of each image during the training. Indeed, the CycleGAN uses a batch size of 1 which means that we can get the loss for each image during the training process. As said before, there are 3 types of loss : adversarial loss (G_A loss and G_B loss), cycle consistency loss (Cycle_A loss and Cycle_B loss) and identity loss (Idt_A loss and Idt_B loss). In order to find which loss is the best to find outliers, we trained a CycleGAN using a dataset with a majority of good images but with obvious bad outliers outside the domain.

For instance, we trained a CycleGAN with 73 day images (for domain A) and 73 fog images + 5 nights images (for domain B). At each epoch, we calculate the average cumulative loss of each image (for every type of loss) and rank images according to these losses. A perfect system would rank the 5 outliers (night images here) first. The table (a) of Table 5.1 shows the ranking of images according to their losses at epoch 10. It shows that the cycle consistency of B is the best measure to find outliers.

We chose epoch number 10 because experiments have shown that it is at this epoch that

Table 5.1 Ranking of images according to their loss. The outliers are colored in red while the normal images are colored in green. We see that Cycle_B is the loss that is the most useful to find outliers.

Rank	G_A	G_B	Cycle_A	Cycle_B	Idt_A	Idt_B
Rank 1	0.87	1.29	2.54	6.63	2.56	1.25
Rank 2	0.59	0.98	2.23	5.76	2.51	1.16
Rank 3	0.58	0.77	2.17	5.02	2.13	1.13
Rank 4	0.57	0.71	2.13	4.72	1.96	1.07
Rank 5	0.57	0.65	2.13	4.59	1.72	1.06
Rank 6	0.55	0.56	2.12	3.14	1.69	1.04
Rank 7	0.53	0.54	2.07	3.01	1.58	1.04
Rank 8	0.52	0.49	2.07	2.85	1.54	0.99
Rank 9	0.52	0.49	2.02	2.79	1.49	0.99
Rank 10	0.51	0.47	2.01	2.72	1.49	0.99

(a) Obvious outliers

Rank	Loss Cycle_B
Rank 1	3.81
Rank 2	2.93
Rank 3	2.82
Rank 4	2.72
Rank 5	2.69
Rank 6	2.59
Rank 7	2.51
Rank 8	2.46
Rank 9	2.46
Rank 10	2.42

(b) Fog outliers

Rank	Loss Cycle_B
Rank 1	2.06
Rank 2	2.04
Rank 3	1.97
Rank 4	1.81
Rank 5	1.78
Rank 6	1.73
Rank 7	1.72
Rank 8	1.69
Rank 9	1.68
Rank 10	1.65

(c) Night outliers

we find the biggest discrepancies between outliers and normal images for Cycle_B. We did the same experiment with a CycleGAN trained with a modified version of Dataset 2 (from §5.1) 1350 day images (domain A) and 1350 night images + 8 fog images (domain B). The method found the 8 outliers with a bigger margin at epoch 10. It means that this method works very well when outliers represent a tiny part of the dataset.

Now that we have an indicator to find outliers that will harm the training of CycleGAN, we tried to use the loss indicator on more subtle problems. The previous outliers were "obvious". But we also tried the method to solve the data engineering of Fog dataset. In fact, "Fog" is not a uniform domain, and there are many types of fog. Experiments have shown that images with a heavy fog (for instance, when the fog covers the whole image) hamper the training of the CycleGAN. That is why we did an experiment where we used a dataset of 73 day images (domain A) and 73 medium fog + 5 heavy fog (domain B). This experiment is interesting because there is no discontinuity between "heavy fog" and "medium fog" and then the process of data engineering is subjective. The table (b) of Table 5.1 shows the result of this experiment. We see that the method is less efficient on subtle data engineering but still shows good results.

4 out of 5 outliers are in the top 10 of images by loss values. However, the margin between outliers and normal images is smaller than with obvious outliers.

We also tried this method on the night dataset. In fact, the night dataset had images with a night label that can be considered as obvious outliers. These images can be images taken during the evening (at the limit between day and night) or images with an unusual purple light from ambulances. These images were hampering the training of the CycleGAN and we saw the effects of these outliers on the converted images in §5.1. That is why we tried this method to find these "outliers". We used a modified version of Dataset 2 (from §5.1) : 1350 day images (domain A) and 1350 normal night images + 10 unusual night images (domain B). We can see the results in the table (c) from Table 5.1.

These experiments show that this method has serious limitations when it comes to finding subtle outliers. Furthermore, the results from this method depend on the initialization of the CycleGAN and then we can have different results with different initializations. That is for this reason that this method has not been used to train the CycleGANs that have been used in §8. However, the initial results with obvious outliers were promising and this method could be enhanced in the future.

Chapter 6

COPING WITH HIGH FREQUENCIES

The CycleGAN is not a perfect method to convert an image from one domain to another domain. That is why we tried to explore whether postprocessing could help to enhance AMOEBA.

6.1 High frequencies induced by CycleGAN

The training of the CycleGAN involves reducing the Cycle Consistency loss $\|F(G(x)) - x\|$. The problem is when one domain (day for example) has more details than the other (night) and then the CycleGAN needs to find a trick to reconstruct the original image. We know that CycleGAN uses high frequencies of the image to hide information that are useful for cycle consistency [4]. That's how CycleGAN can retrieve the original image from a domain that has less information than the original domain. Humans can't see these high frequencies but they still exist in the picture. Figure 6.1 shows how complex details like trees, clouds and buildings can be hidden in an image that has almost no visible details.

We also know that high frequencies can be problematic for image recognition algorithms using deep learning, these issues are called "adversarial attacks" [10]. That is why we can call these high frequencies induced by the training of the CycleGAN as "self-adversarial attacks" [2]. Our goal is to limit the effects of these high frequencies by applying postprocessing to images before using them to train deep learning models. The Figure 6.2 presents an overview of our updated system.

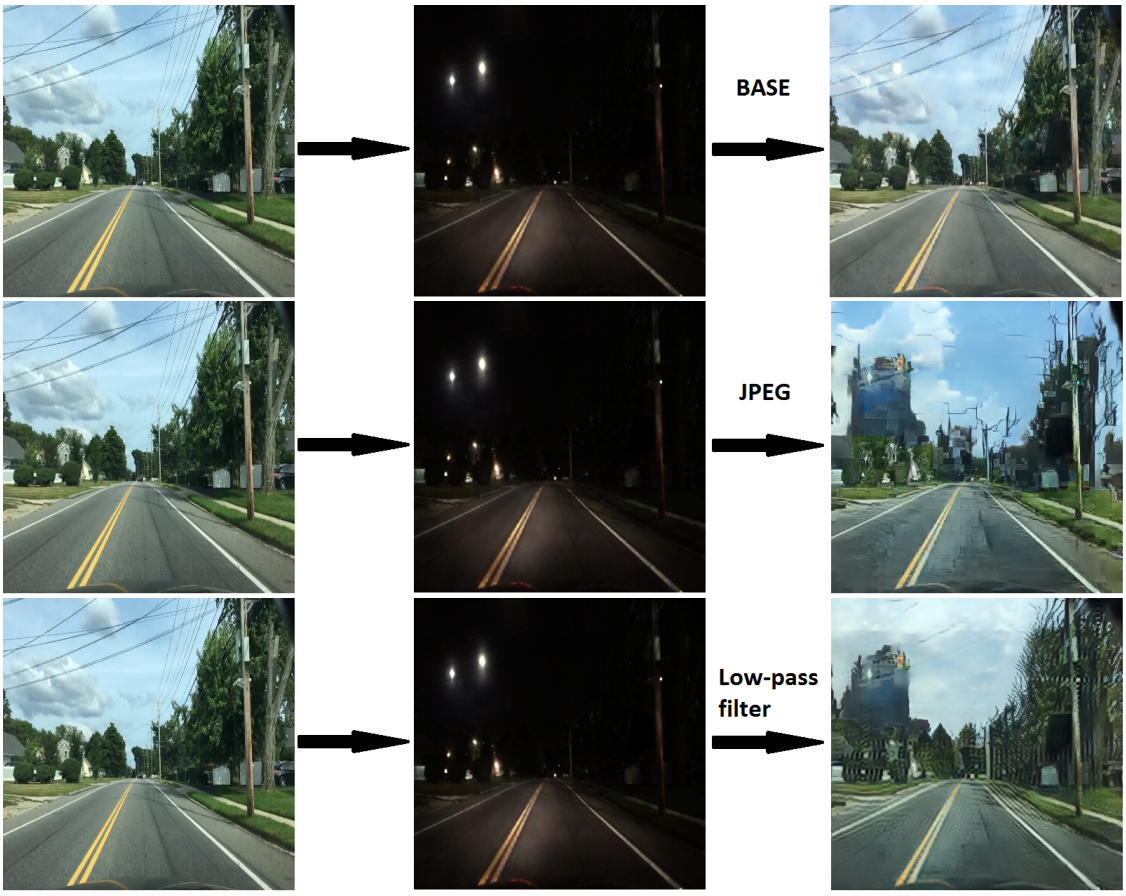


Figure 6.1 Cycle consistency. The CycleGAN can guess what was the original image. However, we can break this consistency with compression or suppression of high frequencies.

6.2 JPEG compression

JPEG compression is a lossy compression for images. The principle of JPEG is that humans can't see high frequencies of an image and then these frequencies are less critical than low frequencies. Then JPEG compression reduces the amount of information used to encode the high frequencies of an image (this process is called "quantization"). Studies[1] have shown that JPEG compression can mitigate the effects of adversarial attacks. The compression of an image can be controlled by a quality coefficient Q that goes from 0 (the most compressed) to 100 (least compressed).

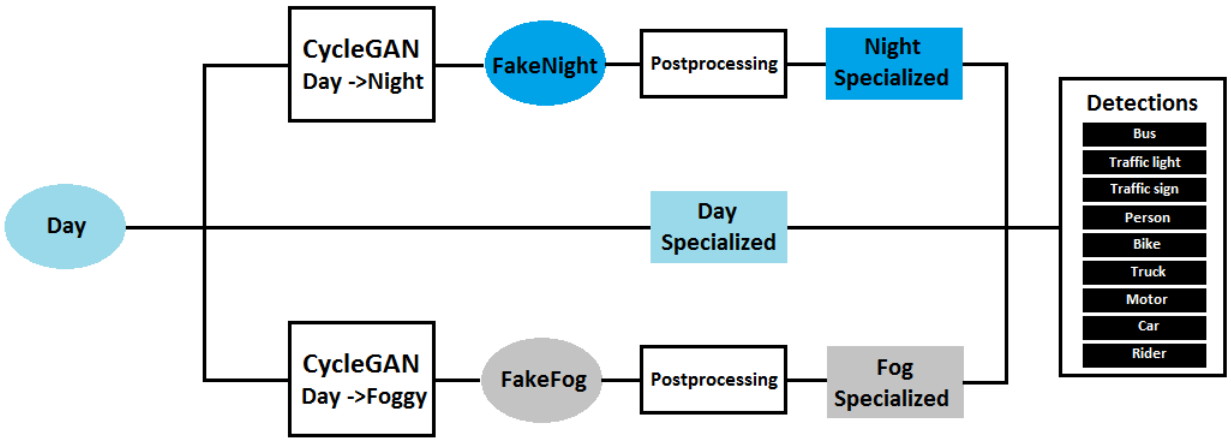


Figure 6.2 Coping with self adversarial attacks. We postprocess the converted images to increase the performance.

That is why we tried to apply a JPEG compression to converted images before using them for training object detection models. We trained different YOLO models with images that passed through different compression quality to see whether or not alleviating these high frequencies can enable better results. We can see in the Figure 6.1 that the JPEG compression breaks the cycle consistency. We can note that shapes that look like squares appear in the converted image. It seems to reflect the splitting of the images into 8x8 blocks (JPEG compression algorithm). As a result, new edges appear and they can form squares after passing through the CycleGAN. A fake building that doesn't exist in the original image appears on the left on the converted image. It shows that a small change on these high frequencies can have tremendous effects on the cycle consistency.

6.3 Supressing high frequencies

Another technique to prevent adversarial attacks is simply suppressing high frequencies of a picture [34]. The first step is to convert the image into frequency domain through a discrete fourier transform (DFT). We note x the original image and y the same image in the frequency

domain. The conversion is :

$$y_{ij} = \sum_{a=0}^{M-1} \sum_{b=0}^{N-1} x_{ab} e^{-j2\pi(\frac{i}{M}a + \frac{j}{N}b)} \tag{6.1}$$

We get a new image y where elements in the center of the image represent low frequencies and elements at the outskirt represent high frequencies. The next step is to suppress the elements at the outskirt by multiplying them by 0 : we multiply the frequency image by a mask with 1s in the center and 0s outside. Then we can apply an inverse DFT to retrieve a new image with only low frequencies components. We use a coefficient R to control the amount of high frequencies that we keep in our picture. R goes from 0 to 208. A low coefficient R means that we suppress most of high frequencies.

We suppress high frequencies of converted images with different R coefficients and we train YOLO models with these new datasets to see the influence of R on the accuracy. The Figure 6.1 tells us that suppressing high frequencies also breaks the cycle consistency of the CycleGAN. It seems that small details like leafs are really affected by this low-pass filter. We can also note that a fake building also appears in the converted image.

Chapter 7

IMPLEMENTATION

In this part, we talk about the details of the implementation. We will explain the datasets used, the architecture of Deep Learning models and the method to measure the accuracy.

7.1 Datasets overview

We worked on 2 different domain adaptation problems : Day to Night and Day to Fog. All images were resized to 416x416.

7.1.1 Day to Night

For the first problem (Day to Night), we only used the BDD dataset [33] as it already had a big number of day and night images (52511 day images and 39986 night images). This dataset has also the advantage of being annotated which is useful for the evaluation part. For the training of cycleGANs, we tested 3 datasets. The first dataset (dataset 1) was randomly chosen in the BDD dataset and had 1750 night images and 1750 day images. The second dataset (dataset 2) was the result of a data engineering from the dataset 1 and had 1350 night images and 1350 day images. The goal of this dataset was to train the CycleGAN on a clean dataset without outliers. However, the data engineering was made by a human and then with subjective criteria (see §5.1).

The training of the first 2 CycleGAN models raised a problem. Indeed, the CycleGAN model

Figure 7.1 Traffic light problem. CycleGANs trained with the first 2 datasets didn't place correctly the traffic lights. That is why we created dataset 3.

couldn't place traffic lights properly. The CycleGAN doesn't seem to be able to detect real traffic lights and place randomly small blue lights in the image.

That is why we created the third dataset (Dataset 3) which was the result of a data engineering from the dataset 2 (800 night images and 800 day images). We removed every image that had a traffic light so that the CycleGAN model don't invent fake traffic light. We can see with Figure 7.1 that training a CycleGAN with the Dataset 3 can prevent the model from creating fake traffic lights. The main drawback is that the model can't place any traffic lights even the real ones.

We used the other images from the BDD dataset for the evaluation part. The remaining day images form the dataset REAL-DAY and the remaining night images form the dataset REAL-NIGHT. These 2 datasets are split to create a training dataset (to train the YOLO models) and a test dataset (to test the YOLO models). To have a consistent dataset, we only picked images taken during good weather condition (without fog or snow). Each time the dataset FAKE-NIGHT is constructed from the conversion of the dataset REAL-DAY so that the comparison between FAKE-NIGHT $\rightarrow$ REAL-NIGHT and REAL-DAY $\rightarrow$ REAL-DAY is relevant.

7.1.2 Day to Fog

For the second problem (Day to Fog), the BDD dataset wasn't sufficient. The BDD dataset has only 130 fog images, and among those images, only 70 are taken during daytime. This number is too low to conduct relevant experiments. That's why we had to gather images from the web (using image search engines) and from the ETHZ fog dataset [23]. We decided to keep the images from ETHZ and from BDD for evaluation as these images were already annotated. We then had a dataset of 100 REAL-FOG images to test our YOLO models (REAL-FOG dataset). We only used the images from the web to train the CycleGAN.

For the training of the first CycleGAN, we used 150 fog images and 150 day images. However, seeing that this first dataset wasn't uniform (some images had a heavy fog while others had a smooth fog), we had to test other datasets by data engineering using the method described in §5.1. After numerous testings, we finally chose a dataset of 73 fog images (medium fog) and 73 day images.

We use the CycleGAN to convert the REAL-DAY dataset into FAKE-FOG dataset and then use this last dataset to train a YOLO model that will be tested on the REAL-FOG dataset. As for the night dataset, we test both REAL-DAY $\rightarrow$ REAL-FOG and FAKE-FOG $\rightarrow$ REAL-FOG.

7.2 Architecture

For the CycleGAN, we used the code from the original paper [35]. The chosen generator was resnet9blocks [12] and the discriminator was the PathGAN. For the day/night conversion, the initial learning rate was 0.0002 and it kept this value during the first 100 epochs, then it was linearly decayed to 0 during the last 100 epochs as in the original paper [35].

For the day/fog conversion, the initial learning rate was 0.0001 and it kept this value during the first 100 epochs, then it was linearly decayed to 0 during the last 200 epochs. This adjustment was necessary because the CycleGAN had to train longer with a lower learning rate to have the most realistic fog possible.

Table 7.1 Class representation. The different classes are not equally represented in the dataset.

Class	AP
Bus	0.148
Traffic light	0.151
Traffic sign	0.339
Person	0.26
Bike	0.098
Truck	0.242
Motor	0.044
Car	0.431
Rider	0.076

(a) Example of results

Class	Gross number	Fraction
Bus	7844	0.011
Traffic light	128034	0.182
Traffic sign	83148	0.118
Person	60887	0.086
Bike	4998	0.007
Truck	20473	0.029
Motor	2149	0.003
Car	392492	0.558
Rider	3314	0.004

(b) Fraction of each class

The YOLO model chosen is YOLOv3 which is an improvement of the original YOLO [20]. We used a batch size of 8, a number of epochs of 100.

The GPU used is an NVIDIA Quadro P6000.

7.3 Measuring the accuracy

For each YOLO model, we keep the model that has the highest overall accuracy. Then, we need to define an overall accuracy score. The problem is that each test on the dataset gives a result in the form of an array like the table (a) of the Table 7.1 with the Average Precision (AP) for each object class.

The first idea to define an overall score would be to calculate the average AP for all classes :

$$\overline{mAP} = \frac{1}{N} \sum_{k=1}^{N} AP[k]. \tag{7.1}$$

The issue with this method is that classes are not equally represented in the BDD dataset. We calculated the fraction of each class in the BDD dataset for day images. As shown in the

table (b) of the Table 7.1, some classes are more common than others, there are 392492 cars whereas only 2149 motors. It means that the YOLO model spends the majority of his time on specific objects.

That is why we choose to calculate the overall accuracy with weights representing the fraction of each class :

$$\overline{mAP} = \frac{1}{N} \sum_{k=1}^{N} w[k] AP[k]. \tag{7.2}$$

Where each w[k] represents a fraction of a particular class. The classes that have the lowest fractions are those who have the highest variations during the training, which is normal as the model doesn't spend a lot of time on them. This way of calculating the overall accuracy helps us to get some stability and give more weights to more relevant classes.

Chapter 8

EVALUATION

In this section, we sum up all the results that we got from our experiments.

8.1 Data Engineering

We tried to train CycleGAN models with 3 different datasets (see §7.1.1). We produced 3 different FAKE-NIGHT datasets and trained 3 different YOLO models. We eventually tested these YOLO models on the testing dataset (REAL-NIGHT images) and measured the AP for each class. The results described in Table 8.1 show that there is a slight improvement between Dataset 1 to Dataset 2. It seems that the artifacts created during the training of the dataset 1 have a negative outcome on the YOLO model. A data engineering makes sense because some images are too different from others and then prevent the CycleGAN from learning a clear and realistic conversion.

However, we don't see any improvement with the Dataset 3. In fact, we have a worse mAP with Dataset 3 than with Dataset 2. We tried to suppress all traffic lights to prevent fake traffic lights from appearing in the converted images. The problem is that REAL-NIGHT images have traffic lights and then the YOLO model needs to get used with these objects to work correctly. For the rest of the thesis, we continued our experiments with the FAKE-NIGHT dataset obtained with the Dataset 2.

Table 8.1 Accuracies for different experiments of FAKE-NIGHT → REAL-NIGHT compared to the baseline DA. The YOLO models have been trained on different datasets produced by 3 different CycleGAN. The first 3 columns represent the results of AMOEBA (FAKE-NIGHT → REAL-NIGHT) and the last column represents the baseline DA (REAL-DAY → REAL-NIGHT).

Class	Dataset 1 (1750)	Dataset 2 (1350)	Dataset 3 (800)	Baseline DA
Bus	0.143	0.169	0.16	0.123
Traffic light	0.14	0.148	0.138	0.273
Traffic sign	0.353	0.351	0.334	0.412
Person	0.265	0.267	0.258	0.278
Bike	0.112	0.122	0.129	0.163
Truck	0.218	0.267	0.233	0.225
Motor	0.037	0.036	0.012	0.055
Car	0.431	0.441	0.432	0.443
Rider	0.083	0.11	0.087	0.091
Overall	0.38	0.39	0.38	0.40

8.2 Ignoring certain class

Some objects are not correctly converted by the CycleGAN. To illustrate this problem we explore the case of traffic lights. We saw that traffic light is the class that brings about most of the problems with the YOLO models. Indeed, the CycleGAN can't place properly real traffic lights and invents fake traffic lights that are placed randomly in the converted image.

This class has a low AP with AMOEBA (0.148) compared with the baseline DA (0.273) and is well represented in the BDD dataset. Then the YOLO model tries to correct this error during its training phase and it affects the training of other classes. That is why we decided to train a YOLO model on the FAKE-NIGHT dataset without taking into consideration the traffic lights. We suppressed all annotations about traffics lights in the labels and measured the accuracy of the YOLO model on the REAL-NIGHT dataset. Table 8.2 shows us that suppressing traffic lights from the annotations has a positive impact on the training of other classes. It corroborates our

Table 8.2 Effect of ignoring traffic lights. Ignoring traffic lights increases the accuracy of the YOLO model on other classes.

Class	8 classes	9 classes	Difference
Bus	0.161	0.169	-0.008
Traffic light	N/A	0.148	N/A
Traffic sign	0.371	0.351	0.02
Person	0.269	0.267	0.002
Bike	0.189	0.122	0.067
Truck	0.261	0.267	-0.007
Motor	0.031	0.036	-0.005
Car	0.483	0.441	0.042
Rider	0.114	0.11	0.004
Overall	0.424	0.39	0.034

first assumption that traffic lights had a negative outcome on the training process. However, for the sake of comparison, we also did the same experience for REAL-DAY →REAL-NIGHT and REAL-NIGHT → REAL-NIGHT and summed up the results in Table 8.3. We see that ignoring traffic lights doesn't change much the results of the YOLO model for REAL-DAY →REAL-NIGHT and REAL-NIGHT →REAL-NIGHT. It means that these traffic lights were hampering our system (FAKE-NIGHT →REAL-NIGHT).

These results show us that AMOEBA won't work for certain classes and then we will have to ignore these classes to have correct results. The method to find these classes could be to first test AMOEBA and then compare each class with the result from baseline DA. Then we can define a threshold. If the loss of accuracy of a particular class is above a certain threshold, we discard this class.

Table 8.3 Ignoring traffic lights. (a) represents REAL-DAY → REAL-NIGHT and (b) represents REAL-NIGHT → REAL-NIGHT.

Class	8 classes	9 classes	Difference
Bus	0.114	0.123	-0.009
Traffic light	N/A	0.273	N/A
Traffic sign	0.401	0.412	-0.011
Person	0.288	0.278	0.01
Bike	0.157	0.163	-0.006
Truck	0.204	0.225	-0.021
Motor	0.058	0.055	0.003
Car	0.453	0.443	0.01
Rider	0.126	0.091	0.035
Overall	0.404	0.402	0.002

(a)REAL-DAY

Class	8 classes	9 classes	Difference
Bus	0.189	0.199	-0.01
Traffic light	N/A	0.373	N/A
Traffic sign	0.454	0.427	0.027
Person	0.303	0.324	-0.021
Bike	0.241	0.234	-0.034
Truck	0.295	0.303	-0.008
Motor	0.123	0.112	0.003
Car	0.539	0.547	-0.008
Rider	0.155	0.134	0.021
Overall	0.48	0.484	-0.004

(b) REAL-NIGHT

8.3 Coping with high frequencies

8.3.1 Errors and relation with the size

The size of objects is essential for Deep Learning models. That is the reason why we measured the average size of each class in the BDD dataset. This measurement is pretty simple as we consider objects as rectangular boxes.

We then compared this size with the accuracies of the YOLO model trained on the REAL-DAY dataset and a YOLO model trained on a FAKE-NIGHT dataset (Dataset 2), both tested on a REAL-NIGHT dataset. We can see a trend from Table 8.4, in fact the 4 classes that are the smallest are also the classes that don't get any improvement from AMOEBA whereas the 4 biggest classes experience a boost. This is a trend that also has been seen when we considered 9 classes.

We know that the high frequencies of an image represent small objects and then we can think that the high frequencies induced by the CycleGAN will have bad effects on small objects.

Table 8.4 Effect of object size on accuracy. The size of objects has a big influence on the accuracy score.

Class	FAKE-NIGHT →REAL-NIGHT	REAL-DAY →REAL-NIGHT	Difference	Object size
Bus	0.161	0.114	0.047	47.6
Truck	0.261	0.204	0.057	34.9
Car	0.483	0.453	0.03	10.3
Bike	0.189	0.157	0.032	8.1
Motor	0.031	0.058	-0.027	7.95
Rider	0.114	0.126	-0.012	7.53
Person	0.269	0.288	-0.019	3.5
Traffic sign	0.371	0.401	-0.03	1.29
Overall	0.424	0.404	0.02	NA

However, we should be careful about these results as the class imbalance in the dataset and the randomness of initialization of the network could also explain these results.

8.3.2 Effect of JPEG compression

We studied the effect of JPEG compression for 3 different qualities : 40, 60 and 80. The Figure 8.1 shows no clear improvement when we use the JPEG compression. It seems that we get the worst result when the quality is lower (40). One explanation could be that the JPEG compression produces new edges (because the algorithm divides the image into 8x8 blocks). These edges can have bad effects on the YOLO model which, explain, our results.

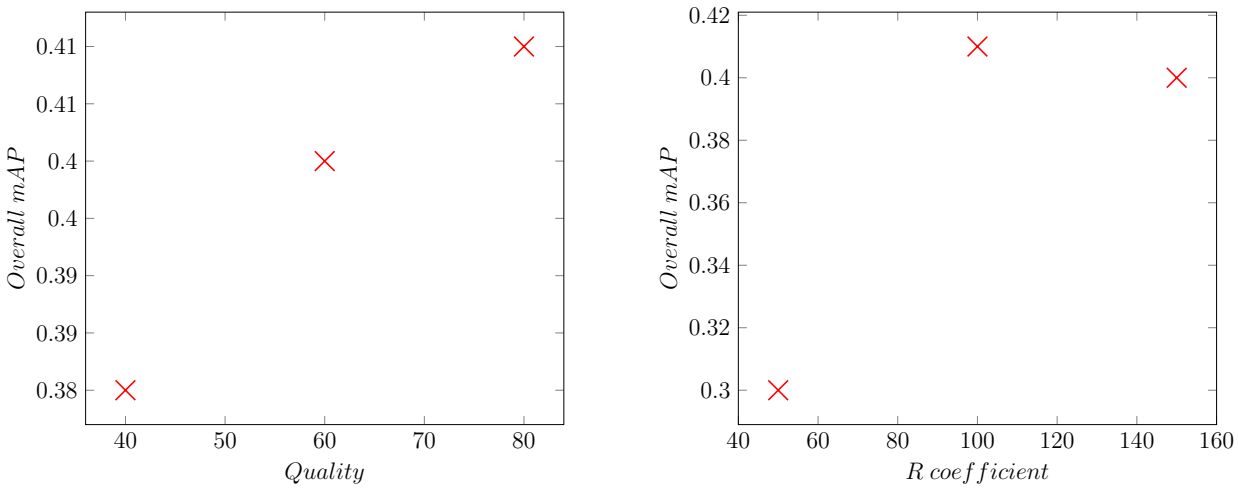


Figure 8.1 Effect of dealing with high frequencies. Jpeg compression (left graph) or suppression of high frequencies (right graph) don't seem to be effective to increase the mAP score.

8.3.3 Effect of suppressing high frequencies

We also tried to study the effect of high frequencies suppression for 3 different R coefficients : 50, 100 and 150 (50 suppress most of the high frequencies). We still don't see any clear improvements and even a steep decline when the R coefficient is too low (50). We still see that the overall accuracy for R=100 is slightly better than the overall accuracy for R=150, but it can be explained by the variance of YOLO model which depends on the initialization.

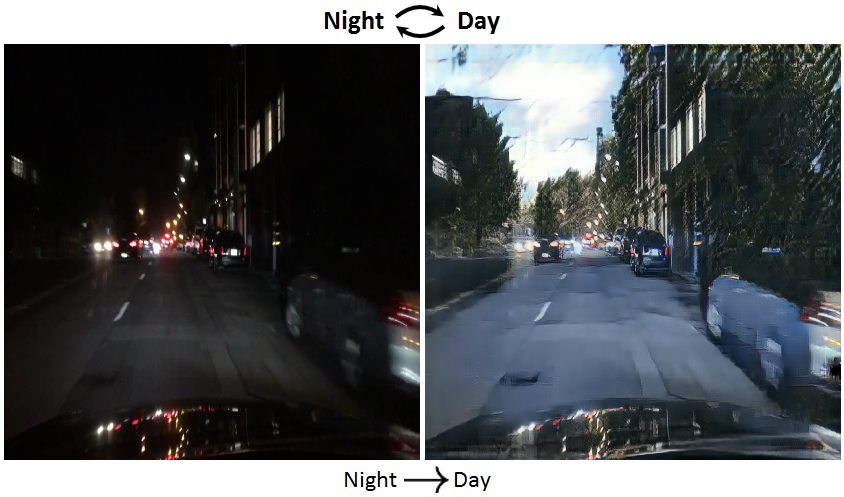


Figure 8.2 Image conversion from night to day. The CycleGAN invent details that don't exist.

8.4 Night to day experiment

Another experiment was to test the domain adaptation from night to day with AMOEBA. This experiment is interesting because the day domain has more information than the night domain. Then, the CycleGAN will have to invent details from a night image. We can see an example from the Figure 8.2 where the CycleGAN struggles to differentiate a building and a tree. This problem appears in many images and it seems that the CycleGAN invents trees that don't exist (these trees are also unrealistic).

Table 8.5 Results from night to day. AMOEBA doesn't work in the direction from night to day.

Class	FAKE-NIGHT →REAL-DAY	REAL-NIGHT →REAL-DAY	Difference	REAL-DAY →REAL-DAY
Bus	0.092	0.075	0.017	0.283
Traffic sign	0.226	0.285	-0.059	0.481
Person	0.222	0.274	-0.052	0.378
Bike	0.128	0.128	0	0.317
Truck	0.191	0.186	0.005	0.332
Motor	0.019	0.029	-0.01	0.204
Car	0.497	0.513	-0.016	0.596
Rider	0.063	0.082	-0.019	0.194
Overall	0.392	0.42	-0.028	0.532

We note from the Table 8.5 that AMOEBA (FAKE-DAY → REAL-DAY) gives us a worse overall mAP than the baseline DA (REAL-NIGHT →REAL-DAY). It shows us that the fake details that our cycleGAN produces are not realistic enough to mimic a real day environment. Then the YOLO models won't be able to work efficiently on a REAL-DAY dataset. Then the assumption that AMOEBA only works when the original domain has more information than the target domain seems correct.

8.5 Fog dataset

The conversion from day to fog goes in the direction of a loss of information. Then it makes sense to try AMOEBA for this specific domain adaptation. The problem was the availability of datasets to perform correct experiments. We still tried AMOEBA with the images we could gather.

The data engineering part of the fog dataset was painstaking. "Fog" is not a uniform domain, there are many types of fog. An image can have a heavy fog that covers the whole picture or can have a shallow fog that covers the horizon. We decided to take pictures representing

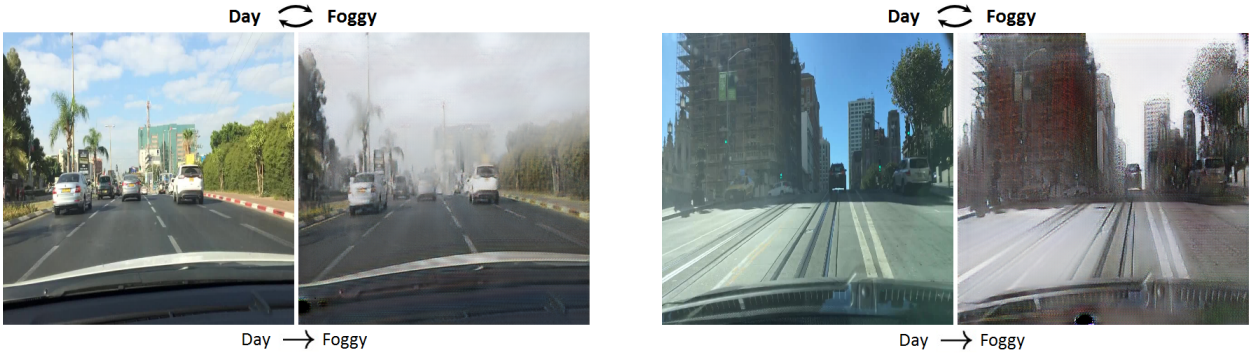


Figure 8.3 Image conversion from day to fog. The CycleGAN is trained with a small cherrypicked dataset of fog images.

"medium" fog. In a bid to have the most uniform dataset for the training of the CycleGAN, we only had 73 fog images (chosen by a human) and 73 day images. 5 We also had to tune the CycleGAN (decrease the learning rate) to have a smooth fog.

The CycleGAN was applied to the same day dataset used to produce FAKE-NIGHT images. The converted images were not as realistic as for the night domain, but we still had some good conversions.

As we can see on Figure 8.3, there are successful cases like the image on the left where the CycleGAN understands that the fog is a function of the depth of the image. This type of image already exists in the training dataset, that is why we can have successful images in these cases.

However, the image on the right represents a failure of our CycleGAN. This type of image doesn't exist in the training dataset. The CycleGAN doesn't know how to cope with the shades and the luminosity. As a result, we have an unrealistic black and white picture.

To measure the performance of AMOEBA, we only took into account 4 classes (bus, person, truck, car) because other classes weren't enough represented. The results from Table 8.6 shows us that AMOEBA is not better than the baseline DA for the fog dataset. As said before, there were much more failures for this domain conversion than with the night dataset. Also, we didn't have a dataset large enough so that the CycleGAN knows how to handle many different

Table 8.6 Results of AMOEBA on REAL-FOG dataset. AMOEBA doesn't work on the fog dataset.

Class	FAKE-FOG →REAL-FOG	REAL-DAY →REAL-FOG	Difference
Bus	0.384	0.417	-0.033
Person	0.369	0.421	-0.052
Truck	0.066	0.057	0.009
Car	0.655	0.682	-0.027
Overall	0.594	0.631	-0.037

situations.

8.6 Comparison with another method : UNIT

There are many domain translation algorithm based on GAN. However, the majority of these GANs use a cycle consistency loss (like DualGAN [32] and DiscoGAN [14]) and then are prone to the same problems as CycleGAN (see §6.1).

Another domain translation technique called UNIT [15] (an improvement of Coupled-GAN [16]) is a type of GAN where the cycle consistency is optional. UNIT is based on the assumption that we can build a shared latent space between domains and encode/decode images in this latent space. We tried to use UNIT as a replacement of CycleGAN to see if such a technique can yield better results. We did the same experiments as with CycleGAN : we convert the same true day dataset into a FAKE-NIGHT dataset and train a YOLO model on this new dataset. The dataset used to train UNIT is the same dataset used to train the CycleGAN (Dataset 2). We did 2 different training : one with a cycle consistency loss and another without.

We can see from Figure 8.4 that UNIT produce less realistic images than CycleGAN. Furthermore, we see that when UNIT doesn't use any cycle consistency, it struggles to produce segmented images and we hardly see discontinuities between cars and background. We have more realistic pictures when we take into account cycle consistency.

(a) Original (b) UNIT with CC (c) UNIT without CC (d) CycleGAN

Figure 8.4 Image conversion different techniques. (a) is the original image, (b) and (c) are converted using UNIT with and without Cycle Consistency (CC), (d) is converted using CycleGAN.

Finally, if we see the results when applying the YOLO model in Table 8.7, UNIT is not a better choice than CycleGAN. The overall mAP is lower than with CycleGAN and these results are even worse when we don't use the cycle consistency. It means that even if the cycle consistency bring some problems in the process (see §6.1), it is still the best way to produce realistic images.

Table 8.7 Comparison using UNIT. Replacing CycleGAN with UNIT yield worse results.

Class	CycleGAN	UNIT with CC	UNIT without CC	Base DA
Bus	0.161	0.046	0.019	0.114
Truck	0.261	0.126	0.102	0.204
Car	0.483	0.33	0.225	0.453
Bike	0.189	0.019	0.014	0.157
Motor	0.031	0.001	0.039	0.058
Rider	0.114	0.038	0.016	0.126
Person	0.269	0.10	0.018	0.288
Traffic sign	0.371	0.136	0.045	0.401
Overall	0.424	0.252	0.157	0.404

Chapter 9

DISCUSSION

First we had some limitations during this Thesis. The fog dataset wasn't big enough to do the same experiments done with the night dataset and we had to look for fog images on the internet to cope with this issue. Also, the night dataset had a serious class imbalance (392492 cars vs 2149 motors) which show the limitations of the comparisons between the classes.

AMOEBA can produce realistic converted images in some cases. There is a trade-off when it comes to choosing the dataset to train the CycleGAN. The more diverse the dataset is, the more cases our CycleGAN will be able to handle. We see that with the Fog dataset, there are many cases that we have to handle (*e.g.* different luminosity, shades, closed environment, etc ...) and the CycleGAN will have troubles to handle "exotic" cases. On the other hand, a "diverse" dataset can have harmful consequences on the training of the CycleGAN. Indeed, the CycleGAN needs to learn a clear function with a specific pattern. And we have seen that "exotic" images (even one) can have detrimental effects on the whole training of the CycleGAN. The diversity of the dataset is a trade-off that is hard to find.

The effects of the high frequencies induced by the CycleGAN can be seen with the AP of each individual classes. In fact, AMOEBA struggles to work efficiently with small objects (the objects represented by high frequencies). These high frequencies can be seen as "self adversarial attacks". However, the methods commonly used to fix these issues (JPEG compression, suppressing high frequencies) haven't worked as expected.

Chapter 10

FUTURE WORK

This thesis raises many potential future works. First, the availability of datasets for fog images and the class imbalance for the night images were serious limitations for our experiments. If in the future, we could have a much bigger datasets, and thus we could do much more experiments. We could also test AMOEBA on other domain adaptation problems where one domain has more information than the other.

Also, we could use other methods for the data engineering part of our problem. The data engineering of dataset for the training of the CycleGAN involved humans which are not objective (how do we define "a heavy fog" ?). We tried to develop a method to pick automatically the dataset, but this method has severe limitations, particularly when a domain is not uniform. As said in the §5.1, we need to find representative uniform datasets to train the CycleGAN and there is trade-off regarding the diversity of the dataset.

Another idea to solve this issue would be using an auto-encoder. In fact, an auto-encoder can reduce the information of an image into a low dimension space. This method has proved to be promising to detect differences between domains for images in a concurrent work. Then by having a low dimension latent space that retains most image information, we could apply clustering algorithms to find similar clusters and pick the most representative datasets to train the CycleGAN. We can see an overview of the system in Figure 10.1.

This method could be useful for fog if it can help us to classify between the different types of Fog and then maybe train different CycleGAN for each specific sub domain. This method

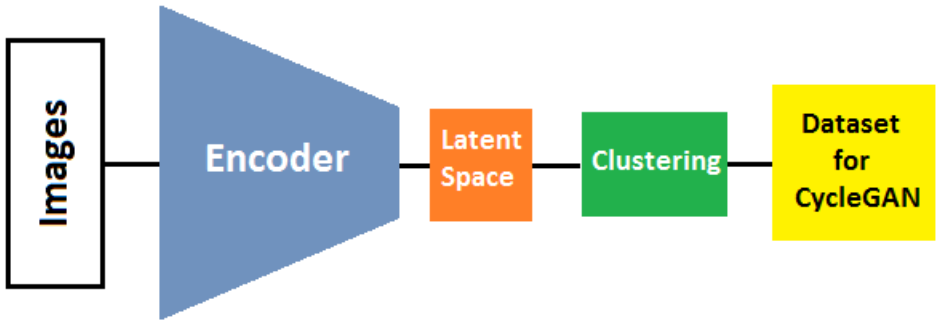


Figure 10.1 Proposal of a new system of Data Engineering. This system would use an auto-encoder and a clustering algorithm to find the most representative dataset of a domain.

could also be helpful for simply find outliers in a dataset like for example photos taken during the evening in a night dataset.

Also, there is space for research on how to mitigate high frequencies induced by CycleGAN. We have shown that these high frequencies have bad outcomes for small objects. This could maybe be fixed by changing the way we train CycleGAN, particularly by changing the way we consider the cycle consistency loss.

Chapter 11

CONCLUSION

In this thesis, we presented AMOEBA, a domain adaptation system based on data augmentation. AMOEBA automatically converts a labeled dataset from the source domain to the target domain using a generative adversarial network. The generated dataset may then be used to train a DL model specialized for the target domain. We found that data engineering improves the efficacy of AMOEBA. A key limitation of AMOEBA is that it does not work well for small objects (*i.e.* high frequencies). Post-processing of images does not address this problem. We plan to address this limitation in future work.

REFERENCES

[1] Ayse Elvan Aydemir, Alptekin Temizel, and Tugba Taskaya Temizel. *The Effects of JPEG and JPEG2000 Compression on Attacks using Adversarial Examples*. 2018. arXiv: 1803.10418 [cs.CV].

[2] Dina Bashkirova, Ben Usman, and Kate Saenko. *Adversarial Self-Defense for Cycle-Consistent GANs*. 2019. arXiv: 1908.01517 [cs.CV].

[3] Konstantinos Bousmalis et al. *Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks*. 2016. arXiv: 1612.05424 [cs.CV].

[4] Casey Chu, Andrey Zhmoginov, and Mark Sandler. *CycleGAN, a Master of Steganography*. 2017. arXiv: 1712.02950 [cs.CV].

[5] Yaroslav Ganin and Victor Lempitsky. *Unsupervised Domain Adaptation by Backpropagation*. 2014. arXiv: 1409.7495 [stat.ML].

[6] Muhammad Ghifary et al. *Deep Reconstruction-Classification Networks for Unsupervised Domain Adaptation*. 2016. arXiv: 1607.03516 [cs.CV].

[7] Muhammad Ghifary et al. *Domain Generalization for Object Recognition with Multi-task Autoencoders*. 2015. arXiv: 1508.07680 [cs.CV].

[8] Ross Girshick. *Fast R-CNN*. 2015. arXiv: 1504.08083 [cs.CV].

[9] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2013. arXiv: 1311.2524 [cs.CV].

[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2014. arXiv: 1412.6572 [stat.ML].

[11] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[12] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[13] Judy Hoffman et al. *CyCADA: Cycle-Consistent Adversarial Domain Adaptation*. 2017. arXiv: 1711.03213 [cs.CV].

[14] Taeksoo Kim et al. *Learning to Discover Cross-Domain Relations with Generative Adversarial Networks*. 2017. arXiv: 1703.05192 [cs.CV].

[15] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. *Unsupervised Image-to-Image Translation Networks*. 2017. arXiv: 1703.00848 [cs.CV].

[16] Ming-Yu Liu and Oncel Tuzel. *Coupled Generative Adversarial Networks*. 2016. arXiv: 1606.07536 [cs.CV].

[17] Mingsheng Long et al. *Learning Transferable Features with Deep Adaptation Networks*. 2015. arXiv: 1502.02791 [cs.LG].

[18] Saeid Motiian et al. *Unified Deep Supervised Domain Adaptation and Generalization*. 2017. arXiv: 1709.10190 [cs.CV].

[19] Xingchao Peng et al. *Fine-to-coarse Knowledge Transfer For Low-Res Image Classification*. 2016. arXiv: 1605.06695 [cs.CV].

[20] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *arXiv* (2018).

[21] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2015. arXiv: 1506.02640 [cs.CV].

[22] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2015. arXiv: 1506.01497 [cs.CV].

[23] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. "Semantic Foggy Scene Understanding with Synthetic Data". In: *International Journal of Computer Vision* 126.9 (Mar. 2018), pp. 973–992. ISSN: 1573-1405. DOI: 10.1007/s11263-018-1072-8. URL: http://dx.doi.org/10.1007/s11263-018-1072-8.

[24] Veit Sandfort et al. *Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks*. Dec. 2019. DOI: 10.1038/s41598-019-52737-x.

[25] Ashish Shrivastava et al. *Learning from Simulated and Unsupervised Images through Adversarial Training*. 2016. arXiv: 1612.07828 [cs.CV].

[26] Eric Tzeng et al. *Adapting Deep Visuomotor Representations with Weak Pairwise Constraints*. 2015. arXiv: 1511.07111 [cs.CV].

[27] Eric Tzeng et al. *Adversarial Discriminative Domain Adaptation*. 2017. arXiv: 1702.05464 [cs.CV].

[28] Eric Tzeng et al. *Deep Domain Confusion: Maximizing for Domain Invariance*. 2014. arXiv: 1412.3474 [cs.CV].

[29] Eric Tzeng et al. *Simultaneous Deep Transfer Across Domains and Tasks*. 2015. arXiv: 1510.02192 [cs.CV].

[30] Mei Wang and Weihong Deng. *Deep Visual Domain Adaptation: A Survey*. 2018. arXiv: 1802.03601 [cs.CV].

[31] Hongliang Yan et al. *Mind the Class Weight Bias: Weighted Maximum Mean Discrepancy for Unsupervised Domain Adaptation*. 2017. arXiv: 1705.00609 [cs.CV].

[32] Zili Yi et al. *DualGAN: Unsupervised Dual Learning for Image-to-Image Translation*. 2017. arXiv: 1704.02510 [cs.CV].

[33] Fisher Yu et al. *BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling*. 2018. arXiv: 1805.04687 [cs.CV].

[34] Zhendong Zhang, Cheolkon Jung, and Xiaolong Liang. *Adversarial Defense by Suppressing High-frequency Components*. 2019. arXiv: 1908.06566 [cs.CV].

[35] Jun-Yan Zhu et al. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. 2017. arXiv: 1703.10593 [cs.CV].